Institutionen för datavetenskap
Department of Computer and Information Science

Final thesis

Investigation and Implementation of a Live Connection between Configura CET and Revit Architecture 2009

by

Freddie Pintar

LIU-IDA/LITH-EX-A--09/008--SE

2009-03-19

Linköpings universitet
SE-581 83 Linköping, Sweden

Linköpings universitet
581 83 Linköping

Institutionen för datavetenskap
Department of Computer and Information Science

Master's Thesis

Investigation and Implementation of a Live Connection between Configura CET® and Revit® Architecture 2009

Freddie Pintar

Reg Nr: LIU-IDA/LITH-EX-A--09/008--SE
Linköping 2009

Supervisor: **Steven Jenkins**
Configura Sverige AB

Examiner: **Christoph Kessler**
IDA, Linköpings universitet

Department of Computer and Information Science
Linköpings universitet
SE-581 83 Linköping, Sweden

Avdelning, Institution
Division, Department

Programming Environments Laboratory
Department of Computer and Information Science
Linköpings universitet
SE-581 83 Linköping, Sweden

Datum
Date

2009-01-31

Språk
Language

☐ Svenska/Swedish
☒ Engelska/English
☐ _____

Rapporttyp
Report category

☐ Licentiatavhandling
☒ Examensarbete
☐ C-uppsats
☐ D-uppsats
☐ Övrig rapport
☐ _____

ISBN
__

ISRN
LIU-IDA/LITH-EX-A--09/008--SE

Serietitel och serienummer **ISSN**
Title of series, numbering __

URL för elektronisk version
http://www.ida.liu.se/ pelab/
http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-ZZZZ

Titel
Title

Undersökning och implementering av en direkt koppling mellan Configura CET® och Revit® Architecture 2009

Investigation and Implementation of a Live Connection between Configura CET® and Revit® Architecture 2009

Författare
Author

Freddie Pintar

Sammanfattning
Abstract

Building Information Modeling -BIM- is an innovative method to seamlessly bridge communication within the architecture, engineering and construction industries. With BIM software you can exchange information during the design, construction, and maintaining. BIM can be seen as a continuation of the CAD software, where the users exchanged information by word of mouth, now is made automatically. To get the effect required for BIM one or more CAD-systems have to work together to exchange information. Revit Architecture is an application by Autodesk where BIM is used from the design and construction to the documentation and maintaining of a building. Configura is one of the major software developers of interior solutions and want to integrate their software with Revit Architecture. The concept of objects in both software system suit well to be used in BIM and with a live connection these could be shared between the applications. One of the conclusions in this investigation was that the only way to have integration between the applications was to use the API provided by Autodesk. And therefore the implementation is limited to the function in it. Revit API is a powerful programming environment that let 3rd party software extend the functionality in Revit. The results show how Remote Procedure Call as a communication tool can be used to exchange data between the applications, how different type of data can be represented in both applications, and why we cannot have a live synchronization.

Nyckelord
Keywords

RPC, Component Software, DLL, Communication, BIM, Revit

Upphovsrätt

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under en längre tid från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

Copyright

The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

© Pintar Freddie

Abstract

Building Information Modeling -BIM- is an innovative method to seamlessly bridge communication within the architecture, engineering and construction industries. With BIM software you can exchange information during the design, construction, and maintaining. BIM can be seen as a continuation of the CAD software, where the users exchanged information by word of mouth, now is made automatically. To get the effect required for BIM one or more CAD-systems have to work together to exchange information. Revit Architecture is an application by Autodesk where BIM is used from the design and construction to the documentation and maintaining of a building. Configura is one of the major software developers of interior solutions and want to integrate their software with Revit Architecture. The concept of objects in both software system suit well to be used in BIM and with a live connection these could be shared between the applications. One of the conclusions in this investigation was that the only way to have integration between the applications was to use the API provided by Autodesk. And therefore the implementation is limited to the function in it. Revit API is a powerful programming environment that let 3rd party software extend the functionality in Revit. The results show how Remote Procedure Call as a communication tool can be used to exchange data between the applications, how different type of data can be represented in both applications, and why we cannot have a live synchronization.

Contents

1 Introduction **1**
- 1.1 About Configura Sverige AB . . . 1
- 1.2 Background . . . 1
- 1.3 Purpose . . . 2
- 1.4 Methodology . . . 2
- 1.5 Discussion . . . 2
- 1.6 Abbreviations . . . 3

2 CET Designer® **5**
- 2.1 Description of CET Designer . . . 5
- 2.2 File Formats . . . 5
  - 2.2.1 CMDRW . . . 5
  - 2.2.2 CM3D . . . 7
- 2.3 Features . . . 7
  - 2.3.1 2D and 3D Environment . . . 7
  - 2.3.2 Alternatives . . . 7
  - 2.3.3 Part Tagging . . . 7
  - 2.3.4 Import/Export DWG . . . 7

3 Revit® Architecture **9**
- 3.1 Description of Revit Architecture . . . 9
- 3.2 Building Information Model (BIM) . . . 9
- 3.3 Elements . . . 9
- 3.4 Category . . . 11
- 3.5 Families . . . 11
  - 3.5.1 System Family . . . 12
  - 3.5.2 In-Place Families . . . 12
  - 3.5.3 Component Families . . . 12
- 3.6 Type . . . 12
  - 3.6.1 Parametric Family Types . . . 12
  - 3.6.2 Nonparametric Family Types . . . 12
- 3.7 Parameters . . . 13
- 3.8 Exporting . . . 13
  - 3.8.1 Exporting to CAD Formats . . . 13

3.8.2 Exporting Animations . . . 13
3.8.3 ODBC Database Table . . . 13
3.8.4 Schedules . . . 14
3.8.5 gbXML . . . 14
3.8.6 Room/Area Reports . . . 14
3.8.7 Industry Foundation Class (IFC) . . . 14
3.8.8 Images . . . 15
3.9 Importing and Linking . . . 15
3.9.1 Linking . . . 15
3.9.2 Importing . . . 15
3.10 Working with Imported Files . . . 15
3.11 Worksharing . . . 16
3.12 Revit File (.rvt) . . . 16
3.13 Revit API . . . 16
3.13.1 External Commands . . . 17
3.13.2 External Applications . . . 17
3.14 Family Editor . . . 17
3.15 Discussion . . . 17

4 Programming Environments 19
4.1 Dynamically Linked library (DLL) . . . 19
4.2 Revit API . . . 19
4.2.1 .NET COM Interfaces . . . 20
4.2.2 Access to Revit Events . . . 20
4.3 Managed Code (C++/CLI) . . . 21
4.3.1 Garbage Collector (GC) . . . 21
4.4 Unmanaged C++ . . . 21
4.5 Configura Magic (CM) . . . 22
4.6 Problems . . . 22

5 Requirements 23
5.1 Discussion . . . 23
5.1.1 Summary . . . 25
5.1.2 Problems . . . 25
5.1.3 Requirements from Configura . . . 25
5.2 GUI Limitations . . . 25
5.3 Start-up and Submit . . . 25
5.4 Data Access . . . 26
5.5 Communication . . . 26
5.6 Synchronization . . . 26
5.7 Object Representation in Revit Architecture . . . 26
5.8 Object Representation in CET Designer . . . 27
5.9 Big Models . . . 27
5.10 Revit 3D Visualization . . . 28
5.11 Object Properties . . . 28
5.12 Manufacturer Meetings - Discussion . . . 29

6 Feature Approach **31**
6.1 Implementation Overview . . . 31
6.2 Communication . . . 31
6.2.1 Remote Procedure Call (RPC) . . . 32
6.3 RPC Protocols . . . 32
6.3.1 Inter-process communication (`ncacn np`) . . . 32
6.3.2 Network Communication (`ncacn ip tcp`) . . . 33
6.4 Object Placement . . . 33
6.5 Parameters . . . 34
6.6 Model Data . . . 34
6.7 File Transfer . . . 35
6.8 Storing Data . . . 35
6.9 Synchronize . . . 35
6.10 User Interface . . . 36

7 Implementation **37**
7.1 Overview . . . 37
7.2 Communication . . . 38
7.3 Interface Definition Language File . . . 39
7.4 File Transfer . . . 40
7.5 Class Mapping . . . 40
7.6 Model Orientation . . . 41
7.7 Synchronization . . . 41
7.8 File Storage . . . 42

8 Measurements **45**
8.1 RPC Transfers . . . 45
8.1.1 RPC Calls . . . 45
8.1.2 File Transfer - Client to Server . . . 45
8.1.3 File Transfer - Server to Client . . . 45

9 Result Analysis **49**
9.1 Introduction . . . 49
9.2 Revit Architecture as Server . . . 49
9.3 Multiple Threads . . . 50
9.4 Object Presentation . . . 50
9.5 Limited RPC Calls . . . 50
9.6 Using serialized Classes . . . 50
9.7 Measurements . . . 51
9.8 Future Improvements . . . 51
9.9 Summary . . . 51
9.9.1 Network Interaction between CET and Revit . . . 51
9.9.2 CET Objects in Revit . . . 52
9.9.3 Revit Objects in CET . . . 52
9.9.4 Examples of Future Projects . . . 52

Bibliography **55**

Chapter 1

Introduction

This first chapter gives an introduction to the thesis. Here it is explained how the idea of the subject of the thesis was developed and for what purpose the thesis is written. Further, the method used when writing the thesis, the delimiting of the problem area, and the structure of the thesis are described.

1.1 About Configura Sverige AB

Configura Sverige AB was established in 1990 and is active today with over 80 employees in both Sweden and the United States. Configura is the market leader in Parametric Graphical Configuration (PGC) technology and their products are used by thousands of people in leading companies around the world.

Configura is a business system to ease the administration of the order process for companies with component based configurable products. When the drawings have been made, the program can extract offer, confirmation of order, and all other information that is needed in the order process.

1.2 Background

The thesis was written as a master thesis of the Master of Science programme in Computer Science, at the Institute of Technology at Linköping University. The idea for the thesis project was developed in cooperation with Steven Jenkins at Configura Sverige AB.

In January 2006 Configura released a product based on a new technology called Configura Extension Technology (CET). [18] CET Designer is a well established software to design environments like Commercial Furniture, Industrial Machinery, Kitchen, and Bath. It helps customers to build and calculate environments easier than other applications. One of its strengths is the support to import AutoCAD drawings and now there have been a request that it also should include integration with another of Autodesk applications, namely Revit Architecture.

Revit Architecture is an Autodesk product. It is a complete building design and documentation system to develop buildings in an efficient way. It follows the BIM (Building Information Modeling) [2] standard that is an innovative method to seamlessly bridge communication within the architecture, engineering, and construction industries. It has built-in functions so that more than one architect can work on the same project at the same time. It also supports many of Autodesk products like AutoCAD and Design Review.

1.3 Purpose

Architects around the world and especially in the United States have gradually started to move over from using AutoCAD to using Revit Architecture. When users of CET Designer want to use drawings from Revit Architecture they have to export them to AutoCAD format which are supported by both products. The problem is that a lot of information is lost when the drawing is exported to AutoCAD format and imported back to Revit Architecture. Therefore Configura need to include something that could help users to work with the same project in both applications without losing much information.

The main task of this thesis will be to find how a bridge between CET Designer and Revit Architecture could be done. Find a way how to take advantage from both application when working on a single project. One way could be to implement support for Revit Architecture's own file format or use an API that let third-party software extend the functionality in Revit Architecture.

From this purpose, we can ask the following questions:

- Is it possible to have a live connection between CET Designer and Revit Arcitecture?
- What data is neccessery to exchange?
- How can we represent this data in both applications?

1.4 Methodology

The authors use a deductive method in this final thesis. This method means that conclusions about individual cases are drawn from available theory [15]. This method suits this project because there are a lot of different approaches available. For example, the different communication protocols availible. To get a good understanding what types of approaches we can use we have to investigate current implementations, an extensive evaluation of different approaches.

1.5 Discussion

Autodesk Developer Network (ADN) is a network that provides information for developers who work with Autodesk products. Through their ADN, Autodesk provides a lot of sources about their applications and their APIs. From Autodesks

webpage we can download a Software Development Kit (SDK) for Revit API. It provides us with lot of documents and examples on how to use the API. Revit Architecture is a highly advanced application and therefore there are both basic and advanced books on how to use it. The implementation uses Windows RPC as a communication tool and there are forums and other webpages explaining how to program with RPC. Especially Microsoft Developer Network (MSDN) has been a good source to understand the RPC programming environment. Both CET Designer and Revit Architecture use approaches like component software, there are many books that explain these approaches. The main focus in this thesis has been to investigate Revit Architecture, the books used during this investigation have been authorized by Autodesk, and by that are very reliable sources.

1.6 Abbreviations

- **.NET** - .NET Framework
- **API** - Application Programming Interface
- **AVI** - Audio Video Interleaved
- **BCA** - Building Commissioning Association
- **BCL** - Base Class Library
- **BIM** - Building Information Modelling
- **C#** - C Sharp
- **CAD** - Computer-Aided Design
- **CET** - Configura Extension Technology
- **CM** - Configura Magic
- **CLI** - Common Language Infrastructure
- **CMDRW** - Configura Magic Drawing
- **CM3D** - Configura Magic 3D
- **COM** - Component Object Model
- **DCE** - Distributed Computing Environment
- **DGN** - Microstation native files
- **DLL** - Dynamic Link Library
- **DWG** - Drawing, AutoCAD file format
- **DXF** - Drawing Exchange Format
- **FBX** - FiLMBOX
- **gbXML** - Green Building XML
- **HTML** - HyperText Markup Language
- **IPC** - Inter Process Communication
- **GC** - Garbage Collector
- **GUI** - Graphical User Interface
- **IDL** - Interface Definition Language
- **IFC** - Industry Foundation Class
- **MSIL** - Microsoft Intermediate Language

- **ODBC** - Open Database Connectivity
- **OSF** - Open Software Environment
- **PGC** - Parametric Graphical Configuration
- **Revit** - Revit Architecture
- **RFA** - Revit Family File
- **RFT** - Revit Family Template File
- **RPC** - Remote Procedure Call
- **RVT** - Revit Model File
- **TXT** - Text File
- **SAT** - Standerd ACIS text files
- **SDK** - Software Development Kit
- **SKP** - SketchUP native files
- **STEP** - STandard for the Exchange of Product model data
- **SQL** - Structured Query Language
- **VB.NET** - Visual Basic .NET
- **XML** - eXtensible Markup Language
- **XREF** - Cross-reference

Chapter 2

CET Designer®

This chapter describes CET Designer.

2.1 Description of CET Designer

CET Designer is used by interior designers when they are preparing a room or area for their customers. The in-build objects have been added by Configura programmers to reflect the real object with all its properties. When the environment has been designed they can extract all information about the objects used in the design. CET Designer has been developed to be used under Windows platforms. Here is an example of a typical user scenario with CET Designer:

> "A customer wants to remodel and gives you a sketch of a room. You prepare a drawing layout in CET Designer. You work simultaneously in 2D and 3D, making it easy to draw and design. The program is aware of the capabilities and limitations of different products, and will not allow you to make mistakes. You make changes quickly and effectively until the customer is satisfied. Together with the drawing, CET Designer has already automatically generated a quote, material specifications and professional 3D images. CET Designer lets your prioritize your customers. Instead of putting in hours on preparing quotes or waiting for someone else to produce drawings, you are with your customers, providing them with optimal solutions on the spot."[3]

Figure 2.1 shows CET Designer with an opened model.

2.2 File Formats

2.2.1 CMDRW

Projects in CET Designer are stored in CMDRW files that have been developed by Configura.

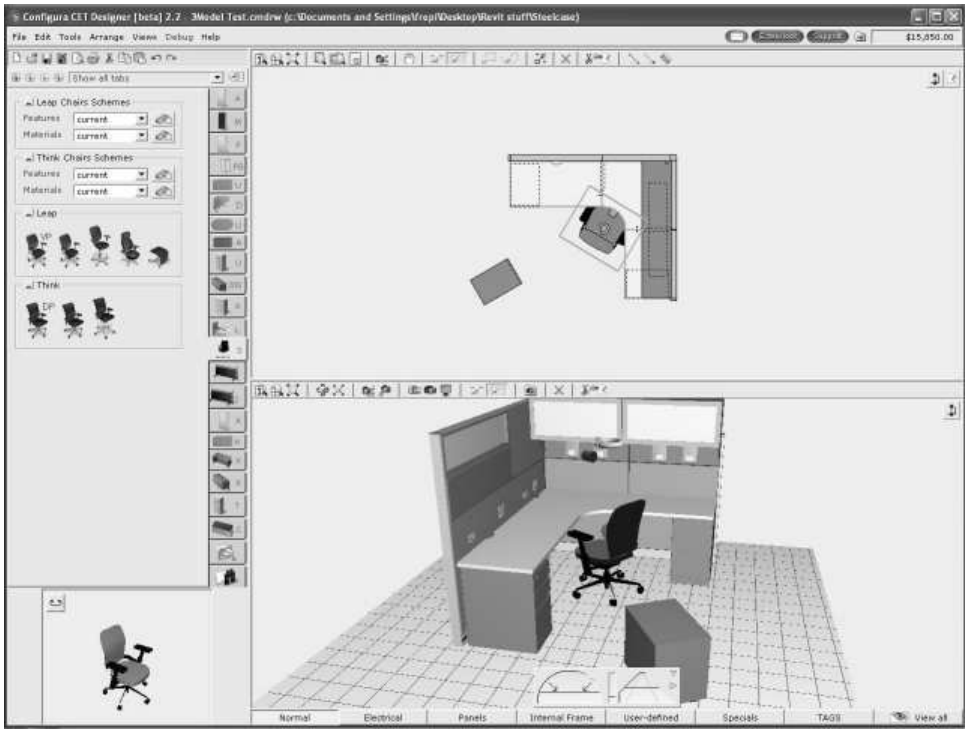

Figure 2.1. Implementation Overview.

2.2.2 CM3D

Objects in CET Designer are called snappers and are stored as CM3D files. This file type has been developed by Configura to meet all requirements of their objects. CM3D files can only be created by Configura.

2.3 Features

2.3.1 2D and 3D Environment

When working in CET Designer there are two different views: A 2D view which is a view from the top and a 3D view that can be used from any direction. We can design in both 2D view and 3D view. When we make changes in one view it show simultaneously in both views.

2.3.2 Alternatives

In CET Designer the user can choose to work with many projects at the same time. With the Alternatives property the user can switch between different projects that work on different workspaces. This is mainly used when the designer want to show different drawings for a customer but could also be used to represent multiple floors of a building.

2.3.3 Part Tagging

Each workspace can be divided into categories. You can mark elements one by one or assign elements by a rectangular marking. Categories are used when a designer like to specify parts of the drawing to be exported to a CAD drawing. This way we can choose to just export a selected part of a drawing.

2.3.4 Import/Export DWG

It is possible to export and import DWG files. When you export a model to DWG you choose to export either 2D or 3D views or both. The DWG links the views so it knows that the 2D view corresponds to the top of the 3D view. When a drawing is exported, there are filters where we can choose which objects should be exported. We can choose to export a selected area or specific categories. When a DWG is exported we have to choose which metric to use and how it should be scaled.

When using the import function we can choose to scale the drawing and also choose what metric to use. When the file is imported we can change color and hide layers in the DWG.

Chapter 3

Revit® Architecture

This chapter describes Revit Architecture.

3.1 Description of Revit Architecture

Revit Architecture is an Autodesk product. It is a complete building design and documentation system to develop buildings in an efficient way. It follows the BIM (Building Information Modelling) standard that is an innovative method to seamlessly bridge communication within the architecture, engineering, and construction industries. It has built-in functions for their file formats so that more than one designer can work on the same project at the same time. It also supports many of Autodesk products like AutoCAD and Design Review. Revit Architecture has been developed to be used under Windows platforms. [16] Figure 3.1 shows Revit Architecture with an opened model.

3.2 Building Information Model (BIM)

Building Information Model is a way to manage information which is created during the whole development of the building. By using BIM, developers can manage information from the first thought to the complete building and even during the maintaining. It is a design methodology that maintains a single database for all information about a building. This is a relatively new term that is used by many of the big CAD companies. [2]

3.3 Elements

Revit Architecture projects use three types of elements: [11]

- **Model elements** - represent the actual 3D geometry of a building. For example, walls, floors, and roofs are model elements.

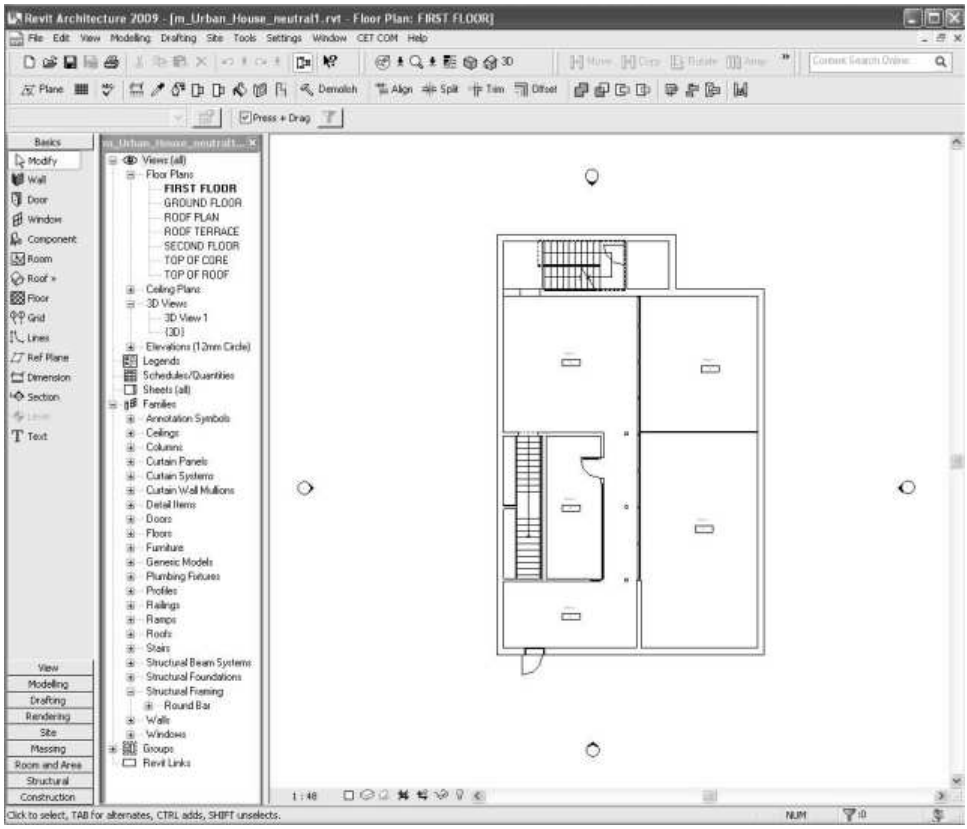

Figure 3.1. Revit Architecture.

- **Annotation elements** - help to document the model. For example, dimensions, text notes, and section tags are annotation elements.

- **Datum elements** - are non-physical items that are used to establish project context. For example, levels, grids, and reference planes are datum elements.

3.4 Category

Categories are organized into families of elements with similar purposes and characteristics. Families are further organized into types, see Figure 3.2. [11]

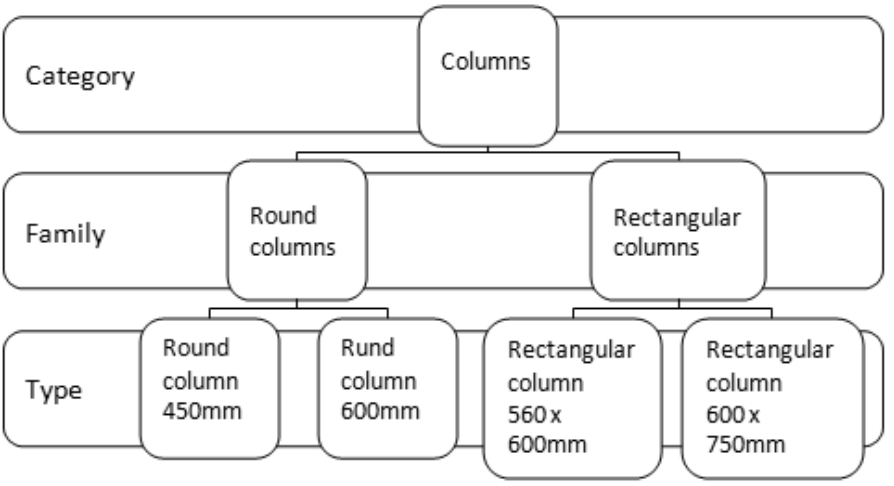

Figure 3.2. Organization of Elements.

3.5 Families

Components and other contents in Revit are referred to as families. Revit Architecture has different types of families which can be edited outside of a project file using the Family Editor. Only those who are used in the model are stored in the project file. Creating families is a critical aspect of working with Revit, because families make up a vast amount of what goes into a model. A family is imported through a Family File (.rfa). Family files can be created in Revit using Family Template Files (.rft). These template file work as a base for the family and it is not possible to create a family without using the Family Template Files. Revit uses three types of families: System familes, In-place families, and Component families. [4]

3.5.1 System Family

System families are created "on the fly" in the Revit project. We can create a new type of system family by duplicating an existing family and changing the preset properties. Below is a list of families that fall into this category. [4]

- Walls
- Roofs
- Stairs
- Ceilings
- Ramps
- Mullions

3.5.2 In-Place Families

In-place families are built in the context of our project using the Family Editor (3.14) interface. The in-place family is good for creating nonstandard element types such as furnitures or other building elements. [4]

3.5.3 Component Families

Component Families are created outside of Revit using the Family Editor. The Family Editor (3.14) can create new elements and modify existing elements. [4]

3.6 Type

When a family have been selected we can choose from different types. The family type specifies the dimensions, material, and a few other characteristics of the element that we can create.

> A type is a member of a family. Each type has specific parameters that are constant for all instances of the type that exist in a model, these are type parameters. Types also have instance parameters, which may vary for each instance of a type in the model. [11]

3.6.1 Parametric Family Types

Parametric families allow variation in size and material. Many types of parametric variations can be stored in a single family as types. An example could be furnitures that has variations in both size and material. [4]

3.6.2 Nonparametric Family Types

Nonparametric families do not provide changeable dimensions. They can only have one type and it is not possible to change the properties of it. Nothing in the family template limits a family to being nonparametric. When Revit imports models from AutoCAD formats, the object represented as a CAD drawing is stored as a nonparametric family type. [4]

3.7 Parameters

Each Revit element has a set of parameters defined by default, such as material, length, area, or other characteristics that describe the element. [4]

3.8 Exporting

Because the building industry is a complex organization with many moving parts and participants it has been supplied with a big collection of export and import functions. Here is a list of types that Revit can export using information from the Revit model. [4]

3.8.1 Exporting to CAD Formats

Revit supports old and new CAD formats. When the export function is used, it exports the current 2D or 3D view. Supported file formats are: [4]

- **DWG** - Files made from AutoCAD, or other applications that can export to this standard format.
- **DXF** - Drawing Exchange Format files. Most software packages write to a .dxf format.
- **DGN** - Microstation native files
- **SAT** - Standerd ACIS text files. Many modelling and fabrication applications can write to this file type.
- **SKP** - SketchUP native files.

If we use the Export command while in a 3D view, Revit Architecture exports the actual 3D model, not a 2D representation of the model. [11]

3.8.2 Exporting Animations

Revit Architecture can create walkthrough animations. These are saved as AVI files.

3.8.3 ODBC Database Table

With the ODBC export function we can export almost all the information embedded in our Revit model to an open ODBC table. Doing so creates a link between our Revit file and another external database such as Excel, Acess, Filemaker, or SQL. ODBC gives us the opportunity to download data from any of the tables in Revit directly to a database using the Microsoft ODBC connector. If we change data in the database it will not reflect to the model inside Revit. [4]

Id	TypeId	PhaseCreated	Volume	Area	Length
253092	55733	82391	48.37521948	145.10752	11.2682625
253131	55733	82391	42.12533063	126.5159561	9.88462028
253155	55733	82391	95.98259325	287.9117908	27.04729615

Table 3.1. Example of a Part from the Walls Table Exported by Revit.

3.8.4 Schedules

We can export information like schedule tables, view lists, material take-offs, key legends, and note blocks as delimited text (.txt format) file that can be read by any spreadsheet/database application. [4]

3.8.5 gbXML

Green Building XML (gbXML) is an XML data type that was created to support the growing trend of sustainability and green building design. gbXML is an export function within Revit that allows us to export specific data about a model for the purpose of performing energy analysis and evaluating building performance. [4]

3.8.6 Room/Area Reports

Room/Area Report is a tool that creates a graphical and mathematical HTML report as a proof of the digital calculation of rooms in our project. The surface of the room is divided into basic geometrical shapes (triangles, rectangles, arc segments), and each shape is described in a table. These reports are required in some authorities in Europe. [4]

3.8.7 Industry Foundation Class (IFC)

IFC support in Revit allows us to export in an industry standard open format that can be used across the building life-cycle. This format can be used by applications that need to work with models generated by other BIM solutions. This is often refereed as intelligent data, which is a way to explain drawings that contain information about its behavior and dependencies. [1]

> "Industry Foundation Classes (IFCs) allow for exchanging of intelligent data between architectural and downstream applications, based on STEP. IFC is a non-proprietary file format that has been recently resurrected as a possible BIM interoperability standard. The goal is to allow the transfer of information between models that have been created using different insert BIM authoring packages. Revit Architecture 2008 supports first-stage IFC 2x3 Certification Import / Export. Revit also has full (second-stage) IFC Singapore Code Checking (BCA) certification, which is export only." [4]

3.8.8 Images

It is possible to export any of the views within Revit Architecture to an image file.

3.9 Importing and Linking

There are two ways to use external files in Revit Architecture: Import and Link.

3.9.1 Linking

If we link a file, any changes made in that original file will be apparent in the Revit file in which it was linked. When someone changes the linked file we can update it in Revit by using refresh in "Manage links..." menu. We cannot explode or modify lines of a linked import. [4]

> "Linking creates a live connection to a file on disk. This allows you to work on the linked file, and then have the Revit model update to reflect the changes in the link. This behaviour is similar to an XREF in AutoCAD" [4]

3.9.2 Importing

An import is not tied to an external file, which will allow us to explode the file and modify the CAD drawing directly in Revit. The explode command allows you to change a grouped item into its individual elements so it can be edited. Once an import has been exploded, the import ceases to exist, and everything becomes lines. These lines are just that: lines, with no inherent intelligence. As lines, we can change their line-type and graphic appearance. [4]

3.10 Working with Imported Files

Revit does not build models based on layers like other CAD packages do. Although layers offer an easy way to control the visibility of objects, they are also the reason that other software packages cannot guarantee quantities. Layers make it possible for an object to be duplicated on many layers and therefore appear more than once in the database. In Revit, every object can exist once and once only. Revit uses categories and subcategories for objects that exist in real life, and uses annotation elements that describe them to control the visibility of what is presented where. When we import file in DWG or other CAD formats in Revit, we can do the following:

- Turn layers on/off.
- Change default color of a layer.
- Explode a DWG file so we can modify/delete elements.

When we explode an imported DWG Revit disassembles the entity as well as any blocks or attributes and simplifies it down to the lowest level of lines, arcs, texts, and hatch (filled regions). [4]

3.11 Worksharing

Revit have features that allow simultaneous access to a model. This is very similar to the linking above but includes the whole Revit project. When we want to build a workset we create a file that will be reachable to everyone who wants to work on the project. Each project member has a copy of the project file on their local computer and can save to a shared project file when they like to. Each team member can lock part of the drawing to prevent other members to edit it.

3.12 Revit File (.rvt)

Revit projects are saved with file extension .rvt. When we save a project using the .rvt file type all information is stored in one file. That includes all families, imported CAD drawings, and images. This file has a tendency to grow in size because the model contain much information about each family object that exist. [4]

3.13 Revit API

Revit Architecture provides an Application Programming Interface (API) that allows extending current functionality of the product. We can define macros that can be saved into the application or project. We can also use it through an external application. For example we can define a macro to add a grid to our project, to rotate a selected object, or to collect information about all rooms in our structure. The Revit API is built into every installation of Revit and enables us to: [9, 11]

- Access all elements in the building information model.
- Query element properties.
- Change element properties.
- Perform additions of some elements
- Locating and extracting Revit content to external files.
- Changing geometry or parameters.
- Creating elements.
- Importing/exporting external file formats.

Revit API has been developed to be used in .NET compliant language including VB.NET, C#, and managed C++. [9]

Revit API supports COM (or ActiveX), which means that the API will work with any language supporting COM (or ActiveX). [1]

A Revit SDK provides us with documentation and code examples in C# and VB.NET.

3.13.1 External Commands

The Autodesk Revit API enables us to add new commands to the user interface of Autodesk Revit. These commands will appear in the "Tools" menu, under a submenu labeled "External Tools". Through the API, external tool commands have access to the Autodesk Revit database, as well as the currently selected elements. [13]

3.13.2 External Applications

The Autodesk Revit API enables us to also add external applications. These applications are invoked during Autodesk Revit startup and shutdown. They can create new top level menus, toolbars, and buttons, as well as react to events occurring in the Autodesk Revit user interface. [13]

3.14 Family Editor

Families are an integral part of working in Revit Architecture, and the key to creating custom content. To create families in Revit we use the family editor.

The Family Editor is a graphical editing mode in Revit Architecture that lets you create and modify the families that you use in your projects. The Family Editor has the same look and feel as the Revit Architecture project environment, but features a single Family Design Bar tab with different commands.

3.15 Discussion

We can start to exclude features in Revit that are not really convenient to us. We want to create a live connection, which means we have to transport data between the two applications. In Basics of the Autodesk Revit Building API [1] they discuss if a developer really need to take benefit of Revit API. Revit has a lot of export functions but the import implementations are very limited.

> "You might not need a custom application using the API if you can work with the exports that are already available from Revit. The ODBC export includes most of the relevant information about the model in separate tables by element type. Unfortunately, the transfer is one way: out of Revit. You cannot subsequently bring data modifications back into your model (without the API)." [1]

There is however one export and import format that have extended functionality. The Industry Foundation Class (IFC) can be used to export whole Revit models to an industrial standard format that can be imported back to Revit. But a Revit model has extensions that the industrial standard does not support, therefore a lot of information will be lost.

The Revit Project file is a very powerful database in its own right. The built in features to work towards a shared Revit Project file could be used to build a live

connection. There is no technical information available neither about the Revit Project file nor worksharing available for us to use.

Revit API gives us a good tool to create a live connection. It has been developed to be used in a .NET compliant language but also supports COM.

> "The Revit API for .NET will be much easier for you to use than COM because it supports both overloaded functions and intelligence in the editor. While the COM version is identical in content, it will be difficult to use during development, and may not be supported in the future. For that reason, I suggest using a .NET language and development tool." [1]

> "Component Object Model is an older set of technologies for application development. While COM is still very prevalent, there are some limitations to it that can only be overcome by using .NET. Microsoft is focusing all future development platforms on .NET and will eventually drop support for COM." [1]

Chapter 4

Programming Environments

This chapter will introduce us to the different programming environments.

4.1 Dynamically Linked library (DLL)

One of the requirements was that the implementation should be made using DLLs. So DLLs will be used in both Revit Architecture and CET Designer to handle the communication.

A dynamically linked library (DLL) is a module that contains functions and data that is located in a external library. DLLs provide a way to modularize applications so that their functionality can be updated and reused more easily. DLLs also help reduce memory overhead when several applications use the same functionality at the same time, because although each application receives its own copy of the DLL data, the applications share the DLL code.

A DLL can contain exported and internal functions. The exported functions are intended to be called by other modules, as well as from within the DLL where they are defined. Internal functions are typically intended to be called only from within the DLL where they are defined. Although a DLL can export data, its data is generally only used by its functions. [6]

4.2 Revit API

The Revit API allows us to program with any .NET compliant language including Visual Basic.NET, C#, and C++/CLI. In addition, the API is focused on integrating analysis and visualization applications into the Revit Building Information Model. All Revit applications including Architecture, Structure, and MEP contain the Revit API. The three APIs are very similar and are jointly referred to as the Revit API. The Revit API is fully accessible by any language compatible with the Microsoft .NET Framework 2.0. [14]

4.2.1 .NET COM Interfaces

Revit API is set up based on Revit Architecture functionality. Revit Architecture is not dependent on the API. Revit API is a class library that only works when Revit Architecture is running. To access Revit Architecture through the Revit API we use interfaces. Specific entry point types are provided in the add-in DLL. These entry point classes implement the external-series interfaces, such as IExternalCommand and IExternalApplication. In this way, the add-in is invoked automatically on certain events or manually from the Tools menu. To load our application we have to create a BCL (Base Class Library) file and add the reference to it in the Revit.ini file. Revit Architecture will then load the DLL file on start up. [14]

- IExternalApplication - The DLL will be executed during the start up. Figure 4.2 show the IExternalApplication interface.
- IExternalCommand - The DLL will be run during an internal command. Figure 4.1 show how IExternalCommand are invoked with Revit Architecture.

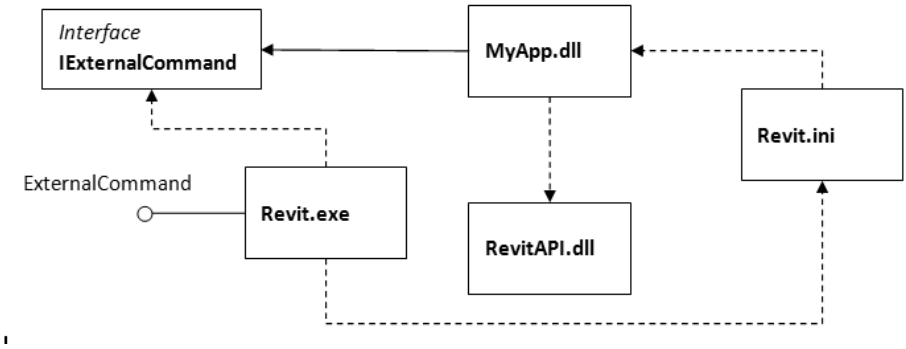

Figure 4.1. Invoking the External Command.[14]

4.2.2 Access to Revit Events

The Revit API lets us access document-specific events. The events available are the following: [14]

- OnDialogBox
- OnDocumentClosed
- OnDocumentNewed
- OnDocumentOpened
- OnDocumentSaved
- OnDocumentSavedAs

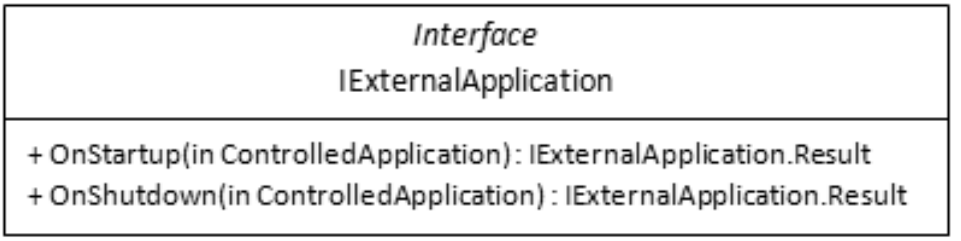

Figure 4.2. IExternalApplication Interface.[14]

4.3 Managed Code (C++/CLI)

The term managed code refers to the Microsoft intermediate language (MSIL) [7] code produced by the compiler. Compiling to managed code is accomplished by use of the /clr compiler option. To use Revit API we have to use a programming language that support Common Language Runtime (CLR). With Managed C++ (CLI/C++) which has support for CLR, the vast majority of existing C++ functions can be compiled to MSIL with no change in semantics. Any C++ program can be compiled to the CLR. However, the runtime defines a particular object model that does not support all features of the C++ language. For example, multiple inheritance of classes is not supported. There are currently no compiler options or pragmas that turn all of a C++ program's classes into Managed Extensions classes. [8]

4.3.1 Garbage Collector (GC)

The .NET Framework's garbage collector manages the allocation and release of memory for our application. Each time we use the new-operator to create an object, the runtime allocates memory for the object from the managed heap. As long as address space is available in the managed heap, the runtime continues to allocate space for new objects. However, memory is not infinite. Eventually the garbage collector must perform a collection in order to free some memory. The garbage collector's optimizing engine determines the best time to perform a collection, based upon the allocations being made. When the garbage collector performs a collection, it checks for objects in the managed heap that are no longer being used by the application and performs the necessary operations to reclaim their memory. [6]

4.4 Unmanaged C++

Unmanaged code is the native machine target code produced by an ordinary compilation. We will use a compiler that produces unmanaged code to create the DLL that will handle the connection from CET Designer to our Revit Architecture

extension.

4.5 Configura Magic (CM)

CET Designer is built on Configura's own programming language called Configura Magic. The programmers have been giving their opinions on the syntax during the development on CET Desinger. The language are compiled to binary code but has support for handeling DLLs. It uses a garbage collector and reflects managed code more than unmanaged code.

4.6 Problems

With the DLL in CET Designer we cannot use common .NET tools. Also C++/CLI creates problems due to the Garbage Collector. The tools we have to use to make an inter-process communication will be imported as native C++ and cannot handle managed C++ pointers. This is because we have to reference to pointers that are located in the Garbage Collector, these pointers can be moved or deleted when the native code are still working with it. We can prevent this by using GCHandle, which provides "handles" into the garbage-collected heap.

Data types used in CM cannot be used in the C++ DLL. Therefore we have to cast types before returning them to CET Designer. However, this will not be a problem between the two DLLs because both use common types.

Chapter 5

Requirements

This chapter will introduce us to an analysis of the features required in the implementation.

5.1 Discussion

We can start talk about why and how users can use a live connection between CET Designer and Revit Architecture. With the current version of CET Designer the developers are focused on one room or floor environment while Revit Architecture is for the whole building. If someone wants to use something from Revit Architecture in CET Designer, she probably wants something she can build a complex but small environment on. And if someone wants to use something from CET Designer in Revit Architecture it will probably be a small but complex environment. A simple conclusion is that borders to one or more rooms will be imported to CET Designer and a complex environment will be exported to Revit Architecture.

There are three types of potential users:

1 **Manufacturers** - Manufacturers would probably want to have their objects to be represented in a Revit Architecture standard with all the properties that their objects have in CET Designer. Their main interest is probably that the designers and architect get what they want.

2 **Designers** - During the design of a room in Revit Architecture using CET Designer the designers would probably like to use the borders which he can build their model on. When the development of the environment is done, they would like to transfer it to one or more objects that can be represented in Revit Architecture. When a designer makes changes to a model that has been transferred to Revit Architecture, the designer should be able to update the representation in Revit Architecture.

3 **Architects** - Architects are interested in the visualization of the rooms. If they have a cost limit they might like to know how much the environment will cost, and if they like to change it they have to contact the designer of

the particular environment. When they want something to be changed, they should be able to update the model after the designers have updated it. The architect might also like to add objects in Revit Architecture that only exist in CET Designer.

Data transferred from Revit Architecture to CET Designer could be information about the rooms and the boarders of the room selected:

- Revit Architecture model data
- Room borders
- Architect comments

These data could be useful to be transferred to Revit Architecture:

- Visualization data
- Developer information
- Cost
- Designer comments

Revit Architecture has objects with properties and it is possible to add a custom property to an object.

The user interface for the live connection could be handled in several ways. Either we could have a live synchronization which means that changes in one application should be reflected to the other. With Revit API this is very hard to achieve because the only events we can catch are document specific events like OPEN, CLOSE, and NEW. This means that we cannot catch when a change is made to a document. We could however make it synchronized one way: When changes are made in CET Designer they reflect directly to Revit Architecture. The easiest way to handle the live connection is to do all transfers from CET Designer because Revit Architecture already has a very well developed interface to control objects from an external application.

Let us consider how the GUI representation can be done. In Revit Architecture we can start our extension in two ways: When Revit Architecture starts (ExternalApplication) and from the menu in Revit Architecture (ExternalCommand). Unfortunately the interface that is used when Revit Architecture starts does not provide all the necessary data which means we have to start our extension from the menu.

Both CET Designer and Revit Architecture have DWG export and import functions. Revit API can only export and import DWG to a file. That means we can transfer DWG files between the applications by storing and loading them on disk. DWGs exported from CET Designer are represented well in Revit Architecture.

5.1.1 Summary

- A full synchronization cannot easily be achieved due to the limitations in Revit API.
- The Revit API extension has to be started from the menu.
- DWGs can be transferred by writing and reading from disk.
- DWGs are represented well in Revit Architecture.
- Cost, Model, and Developer information can be stored in Revit Architecture objects.
- It is not possible to import rendering assets using DWG.

5.1.2 Problems

An easy way to represent complex environments is to use imported DWG in Revit Architecture. The surfaces on a rendered DWG imported from CET Designer have Revit Architecture default rendering properties: All surfaces are gray. We need to know if it is possible to add rendering properties to DWG materials.

Environments imported from CET Designer are often very detailed and can make Revit Architecture very slow.

5.1.3 Requirements from Configura

Configura likes to use a Dynamically Linked Library for the implementation so that the implementation could be used on their other applications. The following is the requirements from Configura:

- Two way communication
- The communication will be between two DLLs.
- The communication can be achieved with or without the users' knowledge.

5.2 GUI Limitations

With the current version of Revit API we are only allowed to make minor changes to the Revit Architecture GUI. We are only allowed to add top menus and top toolbar items.

5.3 Start-up and Submit

When we like to add top menus or top toolbar items we add these through IExternalApplication when Revit Architecture starts. Unfortunately the data that are provided on these extensions include nothing about the model that will be

used later. To get access to the model, the extensions have to be started from the menu. All commands can return three values: Success, Cancel, and Failed. Only when the returning value is success the changes are submitted to the model. This means that the thread has to wait until all changes are made before we can submit. If we modify the model after we returned "success", the model will reset all modifications next time we execute the command.

5.4 Data Access

With the current version of Revit API we can read and write to all elements in a model. There could be a few restrictions but these are only in special cases that probably would not be of our interest. We can delete elements and we can create elements from existing family object. Some document- and program properties are read only. For instance, that includes rendering assets. There are no such limitations in CET Designer.

5.5 Communication

Both applications can appear as server, client, or both. This investigation has focused on local-process communication but it is no problem to implement communication over a network. Revit API does not limit us to add any communication modules. Unfortunately we cannot use IPC extensions that are .NET specific due to the limitations in the DLL used by CET Designer.

5.6 Synchronization

We can not have any live synchronization because the API does not support multiple threads, and with current version of Revit API the only events we can handle are when Revit Architecture closes or opens documents. Only if Autodesk add more events to their API it would be possible to add live synchronization in both ways with good usability. The best way to synchronize is probably by commands.

5.7 Object Representation in Revit Architecture

Elements in Revit Architecture are represented as family object. Family objects can be created using the family editor that is included in Revit Architecture. With the family editor we can create elements with properties and special features. Representing CET object as unique family object would be the superior solution.

With the current version of Revit API we can create instances of existing family objects. There is no API or source code to create new families. If we like to have unique family objects for each object in CET Designer we have to manually create each object using the family editor. [17]

When we import DWGs to Revit Architecture these are stored as a system family instance. With current Revit API we can import but not link DWGs. When

we represent our object using DWGs we could either divide the CET Designer model into multiple DWGs or use one big DWG for the whole model. [14]

It is possible to add DWGs in predefined family instances. Autodesk claim this is not tested and do not guarantee this to be stable. By manually creating empty family instances with rotation and other custom features we can create objects with some of the features that are included in CET Designer objects. Autodesk recomends:

> "Import only one detail at a time so you can take better advantage of Revit's ability to manage sheet referencing. If you have a series of details organized in a single CAD file that you would like to import to Revit, isolate each detail, save it as separate file, and then import." [5]

The author asked Autodesk ADN Support if it is possible to create a new family with custom features and properties that can be used to hold drawing in DWG. The answer was that it is not possible. After some discussion about adding parameters to DWGs, we came to the conclusion that it is possible to add DWGs in existing family instances. That means documentations about the API that explains that it is not possible to create new families are false in some sense.

5.8 Object Representation in CET Designer

With Revit API the DWG export functions are limited to the 2D and 3D views only. This means we cannot represent a room or any other family object in a DWG alone. If we like to export a DWG that can be used in CET Designer it would be a 2D floor view. We can however represent objects in CET Designer with coordinates and other data that are extracted from the family objects. So if we like to import one or more family objects to CET Designer we have to make our own representation using the data we import from Revit Architecture or use a filter to extract them from the exported DWG.

If we add objects in Revit Architecture that already have an object representation in CET Designer we can create a new instance of that object in CET Designer.

5.9 Big Models

We would get best performance if we had a unique family object for all our objects in CET Designer. Revit Architecture can handle many complex objects at the same time but can get very slow depending on the machine it is used on. As mentioned above it is not possible to create own family objects without using the family editor which would have been the best way to do it. Configura is constantly trying to optimize CET objects by reducing the number of polygons and the author does not think this will be any problem.

Another way to improve the speed would be to skin down the DWG by removing elements that are not of interest. We can hide elements in the 2D view only by using the visibility option in Revit Architecture or CET Designer. Elements that

have been marked invisible in CET Designer are also marked invlisible in Revit. Autodesk recommends the following:

> "Don't explode imported CAD files. A CAD file imported into Revit is a collection of objects that is managed as a single entity. If you explode a CAD file, the single object immediately becomes many - and becomes that much more data for Revit to track." [5]

> "If you are importing DWG files, leave them unexploded as much as possible. If you need to hide lines, use the Visibility/Graphic Overrides dialog box to turn layers on and off." [5]

5.10 Revit 3D Visualization

The DWG exported from CET Designer are visualized well in Revit Architecture. Unfortunately the DWG format does not have support for rendering information. When a model is imported to Revit Architecture the materials are represented by a color. With current Revit API we can add rendering assets to the materials but we cannot change the rendering assets that exist in Revit Architecture. Assets are represented in a modified Autodesk FBX format. Autodesk FBX SDK is available on Autodesk's homepage.

Light fixtures are represented as family objects. We can add light fixtures with the current version of the API but as mentioned above, we cannot create new families without the family editor.

The author asked Autodesk ADN Support [17] if it is possible to add own bitmaps to the assets. The answer was that it is not possible at the moment, but that autodesk have logged it as a wish-list item on our behalf.

5.11 Object Properties

It is possible to add properties to objects. Unfortunately we can only create new properties (called types) through the family editor. But this could also be made by creating a shared parameter: If we like to add cost to a room the cost property will be added to all room objects, but can be assigned different values. The cost and developer information will be simple variables and could easily be represented in Revit Architecture objects. It is not possible to assign a shared property to an imported DWG in Revit Architecture. But when we use Revit API we can locate the category of the DWG and use it to add shared parameters. Another way to add parameters would be to import our own family objects with custom properties or add the properties to another element that are related to the imported object.

The author asked Autodesk ADN Support if it is possible to add properties to imported DWGs. The answer was that it is possible:

> "Shared parameters are defined per category, so this means that to attach a shared parameter to your drawing, you will need to keep track of the drawing name and use that to identify the category to define

> the shared parameter for. I have tested this procedure in the attached C# version of the labs and verified that it works." [17]

This will not be a problem if we import DWGs into existing family instances.

5.12 Manufacturer Meetings - Discussion

After phone meetings with manufacturers in the USA there were a few things that came up that have not been thought of.

Many of the manufacturers of interior solutions are releasing own family libraries with their objects in Revit Architecture. This means that, when an architect is working in Revit Architecture, the architect will be able to choose from libraries that are delivered from different manufacturers and use them to get a visual and informational view over the environment. By adding object libraries they are also helping the architects to look even better to their clients.

We have no clear view of how many of the architects actually want to design the interior environment on their own. But the library in Revit Architecture can never replace the objects in CET Designer in complexity and dependence to other objects. The library will only be used to get a draft for the end design. The architects are often working with a limited budget over the whole building and are often only interested in the total cost. So here we talk about budgets that are in the size of millions of dollars. In the end, the cost of the interior solutions is just a few percent of the total cost.

The libraries do not have to be very complex, the important thing is that the architect are choosing them and not their competitors when it comes to the order. If this helps the architects to better develop interior solutions, faster and cheaper, for their clients, it is more likely that they choose the provider of the libraries as their provider of the interior solutions. Whoever gets their libraries used in the model will also have an inside track of easily meeting the specifications for the project if it is put out to bid.

When it comes to the order, the size of the order will be so big that it does not matter if the designer has to work a few extra hours to change the draft design from Revit Architecture to a realistic design in CET Designer.

Chapter 6

Feature Approach

This chapter will introduce how we approach the features required.

6.1 Implementation Overview

The implementation uses two DLL:s. One DLL that is developed in a .NET language that will communicate with Revit Architecture, and one DLL developed in C++ that will communicate with CET Designer. With IPC we can communicate between these DLLs and achieve the coupling we want. Figure 6.1 shows a brief overview of the implementation.

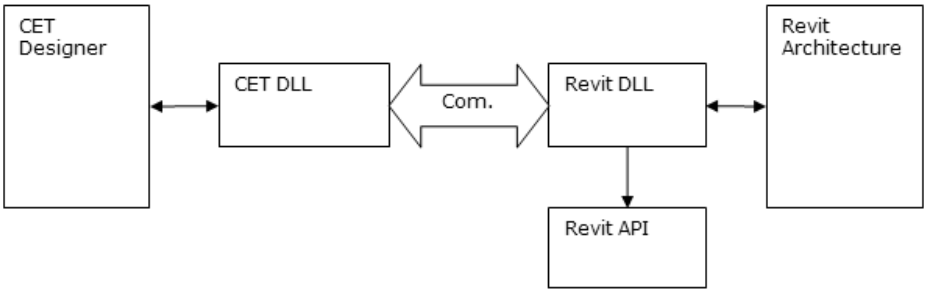

Figure 6.1. Implementation Overview.

6.2 Communication

One of the major features we have to implement is how the applications should communicate. Both applications only work on Windows platforms and therefore we can take benefits by looking on the build-in ways to communicate in Windows.

6.2.1 Remote Procedure Call (RPC)

RPC enables applications to call functions remotely. Therefore, RPC makes IPC as easy as calling a function. RPC operates between processes on a single computer or on different computers on a network. The RPC provided by Windows is compliant with the Open Software Foundation (OSF) Distributed Computing Environment (DCE). This means that applications that use RPC are able to communicate with applications running with other operating systems that support DCE. RPC automatically supports data conversion to account for different hardware architectures and for byte-ordering between dissimilar environments. RPC clients and servers are tightly coupled but still maintain high performance. The system makes extensive use of RPC to facilitate a client/server relationship between different parts of the operating system. A very simple illustration of RPC in client-server communication is shown in Figure 6.2. [6]

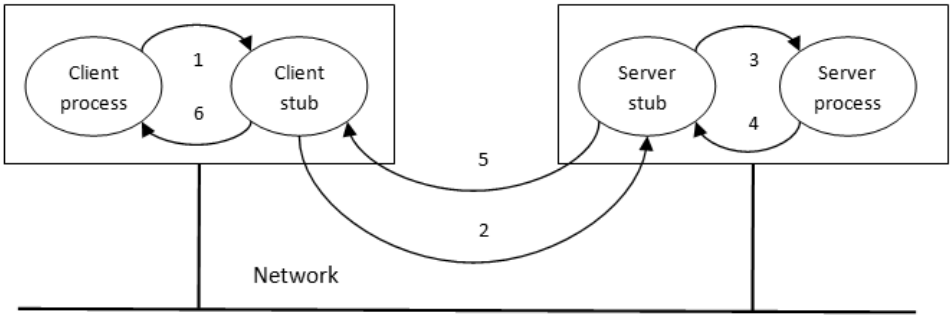

Figure 6.2. RPC Call Procedure.

6.3 RPC Protocols

We can use local communication or network communication with RPC Protocols. When we use local communication we specify the computer name address and when we are going to connect through a LAN or Internet we specify the IP address and port. For inter-process communication we will use the "`ncacn np`" protocol and for network communication we will use "`ncacn ip tcp`".

Table 6.1 shows which protocol sequences could be used with RPC. [6]

6.3.1 Inter-process communication (`ncacn np`)

The "`ncacn np`" keyword identifies named pipes as the protocol family for the endpoint. On Microsoft®Windows 95/98, "`ncacn np`" is supported only for client applications. [6]

Protocol sequence	Network address	Examples
ncacn_nb_tcp	Computer Name	myserver
ncacn_nb_ipx	Computer Name	myserver
ncacn_nb_nb	Computer Name	myserver
ncacn_ip_tcp	Four-octet Internet address, or host name. If the IPv6 network stack is installed, IPv6 is fully supported and an IPv6 address is also accepted.	128.10.2.30 anynode.microsoft.com
ncacn_np	Server Name (leading double backslashes are optional)	myserver \\myotherserver
ncacn_spx	IPX Internet address, or server name	~0000000108002B30612C myserver
ncacn_dnet_nsp	Area and node syntax	4.12
ncacn_at_dsp	Computer name, optionally followed by @ and the AppleTalk zone name. Defaults to @*, the client's zone, if no zone provided	servername@zonename servername
ncacn_vns_spp	StreetTalk server name of the form item@group@organization	printserver@sdkdocs@microsoft
ncacg_mq	Server name	myserver
ncacn_http	Internet address (either four-octet or friendly name, or local server name)	128.10.2.30 somesvr@anywhere.com mylocalsvr
ncadg_ip_udp	Four-octet Internet address, or host name	128.10.2.30 anynode.microsoft.com
ncadg_ipx	IPX Internet address, or server name	~0000000108002B30612C myserver
ncalrpc	Machine name	thismachine

Table 6.1. RPC Protocols.

6.3.2 Network Communication (`ncacn ip tcp`)

The "`ncacn ip tcp`" keyword identifies TCP/IP as the protocol family for the endpoint. [6]

6.4 Object Placement

When objects are transferred between the applications we need to track the location so we can place them correctly in both coordinate systems. We also need to track which object refers to the other application's object. Both CET Designer and Revit Architecture have unique IDs for each object so we can use a stack to store the other objects' ID.

It must be possible to close and open the models but still use be able to synchronize. Therefore we need a way to store data. It is enough to store the data in one of the applications but it is possible to store it in both. In Revit Architecture we have to store it in an external file that we load each time our extension starts. In CET Designer we can store it in the model or in an external file.

Placement is needed during functions like import, export, and move. With a transformation matrix we can send the coordinates for the sending application and transform in the receiving application. CET Designer does not have levels in their model, so we need to know on what level the object should be placed. The problem here is if we have objects that should not be placed on floor level. That could for instance be kitchen lockers or items on tables. Therefore we need another argument to determine the height. The transformation matrix can be sent when the connection is initialized.

The origin of the DWG is always in the center of the CAD object. In CET Designers 2D Space, the CET Objects origin is also in the center, and therefore we can use this origin to be the origin of the DWG. In Revit Architecture we have a few options on how the imported DWG should be placed:

- **Center-to-Center** - aligns the 3D center points of both models.
- **Origin-to-Origin** - places the world origin of the linked file at the Revit Architecture model's origin point. If the linked model was created far from the origin point, linking with this option may put the linked file far away from the main model.

If we place the DWG using Center-to-Center option we can create a transformation matrix of the center points of the model. When we import DWGs or other model data to CET Designer we place it so that it represents a Revit Architecture location.

In case we want to represent each CET Object as its own DWG we can easily place it according to its origin point. One problem is if we transfer multiple CET Objects in one DWG. Then the center point of the DWG will not be represented by any CET Objects origin. This will not be a problem if we can locate the origin of the generated DWG file in both applications.

6.5 Parameters

When we like to specify cost or other information into our model we need to add shared or project parameters in Revit Architecture.

There are a couple of ways to add custom parameters that offer additional information about the element. If we intend to use our parameter in a schedule, we can add the parameter directly in the schedule or add it through project parameters which allows us to add a parameter to a group of elements. If we wish to be able to both schedule and tag our custom parameter, meaning that we wish to see the value of the parameter being called out by the element tag, we need to use shared parameters.

A shared parameter is imported through an external file so that it can be used for many projects. The parameter will be added to a category of elements.

By creating a family object with specified parameters we do not need to use shared and project parameters.

6.6 Model Data

To create a good GUI we need information about the models used and the areas these include. Levels and rooms are things that could be interesting to show in the GUI, and by using the coordinates of these areas we can create a graphical representation to the user.

Coordinates are divided into boundaries. If we like to transfer the coordinates of a room we will get points representing one or more closed loops.

6.7 File Transfer

When a DWG drawing has been created we need to send it to the other application. Therefore we need a way to send files between the applications. Files will be sent when we synchronize between the applications. When we want send a file from CET Designer to Revit Architecture we can use RPC calls to send packages with fixed size. Files that have been created in Revit Architecture from CET Designer can be represented as a unique number.

To send a file from Revit Architecture to CET Designer we can use callback functions. We request a DWG from CET Designer and Revit Architecture use the callback functions to send the file to CET Designer. And when the file is transferred the original function call are returning an index number to an array where the URL or data is located.

6.8 Storing Data

To remember what data we have transferred we need to flush necessary data to the disk. Then we can load it the next time we start a communication. Because the implementation only can be used while the applications are connected, we do only need to store data on one of the computers, the other computer can then request the data. The data that will not be stored in the Revit Architecture project file and that are substantial to us are:

- Mapping of IDs between the objects.
- Mapping between global parameters and the objects in Revit Architecture.
- URLs to the DWGs.

6.9 Synchronize

The synchronization between the applications will be on object level. Each object in CET Designer should be represented by an object in Revit Architecture. Object location, existence of objects, and other data that are within the objects like cost and model id, has to be synchronized too. Data that could be important to compare are:

- Existence of the object
- Location
- Orientation
- Parameters

It is possible to synchronize in both ways. When we synchronize from Revit Architecture to CET Designer we can only apply minor changes because of the limitations in the object representation in Revit Architecture. The user should be able to delete or move objects.

6.10 User Interface

In the first version of the implementation all controls will be done from CET Designer. We need one button to synchronize from CET Designer to Revit Architecture.

Chapter 7

Implementation

In this chapter we will try to explain how the severe problems we run into have been solved. We will be explaining what type of communication we use, and look on the solutions beneath. We go through the interfaces and data transporting methods and explain how the first version of the synchronization has been made.

7.1 Overview

One of the requirements of this project was that the implementation should work so that it can be used by other application. Therefore a DLL was used as an implementation to communicate with Revit Architecture. When building extensions in Revit API we have to create a DLL that is executed through interfaces created in a .NET language. The whole implementation for communication on the client and server ended up in three different languages. In CET Designer we use the language CM. The DLLs uses a combination of only C++ and both C++ and C++/CLI. The choice to use C++/CLI instead of any other .NET language was because it is easier to include C++ code which is used by the RPC libraries.

When the server in Revit Architecture is running the user in CET Designer starts the client from a GUI in CET Designer. The client start to collect information about the models that are open in Revit Architecture and the user chooses to import one or more floor views into CET Designer. These are then represented by DWG drawings in CET Designer. When a floor view has been imported, the user can start to build the environment with CET Designer. The DWG drawing of the floor view is used as a reference for the environment in Revit Architecture. When we wants to transfer the model from CET Designer to Revit Architecture the user start the GUI and choose which floor to export to and clicks on a button to synchronize. The objects are then transported over to Revit Architecture using DWG representations of the objects.

The implementation can only synchronize from CET Designer to Revit Architecture but it also has all features to also synchronize the other way and it is not much of work to include that in the next version. The objects in CET Designer

are the only information in the models that is being synchronized, with the actual orientation and location in the corresponding coordinate systems.

The objects in CET Designer are moved over to Revit Architecture and are represented as imported DWG drawings. One object in CET Designer is represented by one DWG drawing in Revit Architecture. They have a 2D and 3D representation in Revit Architecture and using global parameters. The objects are first transformed to a DWG drawing in CET Designer and then transported with RPC functions. When the DWG drawing has been transferred it can be used to represent one or multiple objects in Revit Architecture.

7.2 Communication

The implementation use RPC to communicate between the server and client. With the current implementation the user can choose to either use Inter-process communication or a communication over network. We are using Microsoft RPC which has been developed to work on Microsoft platforms such as windows XP and Windows Vista which is the operating systems both applications have been developed for. The Microsoft RPC mechanism uses other IPC mechanisms, such as named pipes, NetBIOS, or Winsock, to establish communications between the client and server.

Both CET Designer and Revit Architecture have an implementation to act as both server and client. In current version CET Designer is used as client and Revit Architecture as server. All commands from the user are made from a GUI in CET Designer. Figure 7.1 shows the client and server.

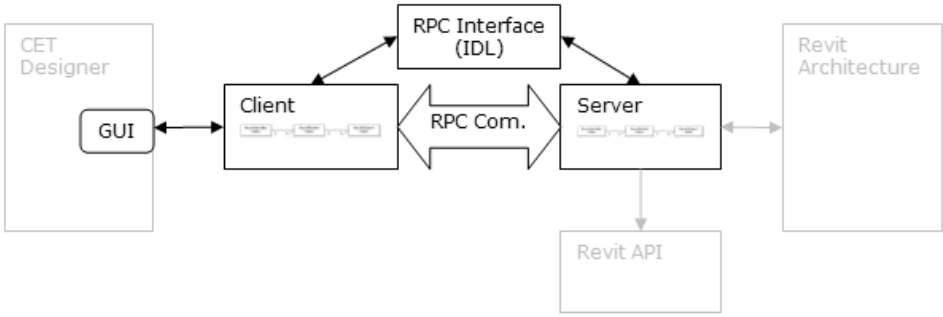

Figure 7.1. RPC Communication.

Microsoft RPC using a compiler called MIDL to generate a common server and client interface. This is compiled from an Interface Definition Language file (IDL) and will generate C++ files for both server and client. The IDL file contains function calls and callbacks. Structures are defined with a fixed size. By using RPC callback functions the server can request or send data to the client. To transport model information between the applications we use function calls specified in the

IDL file.

7.3 Interface Definition Language File

To communicate between the client and server we use a Interface Definition Language file to specify the functions and structures used. Many of the function calls use reference arguments to receive information. The server and client are executed in two different address spaces so we cannot use pointers as reference arguments. The arguments are structures with fixed size in memory.

When the server receives or sends data that can not be recieved from any function call we use function callbacks. These are also specified in the IDL file. The callback functions are used when the server needs information that is essential to the request made by the client or when the server wants to send files to the client. Figure 7.2 show a brief overview of the content of the IDL file. Tables 7.1, 7.2, 7.3 show the structures, calls, and callbacks defined in the IDL file.

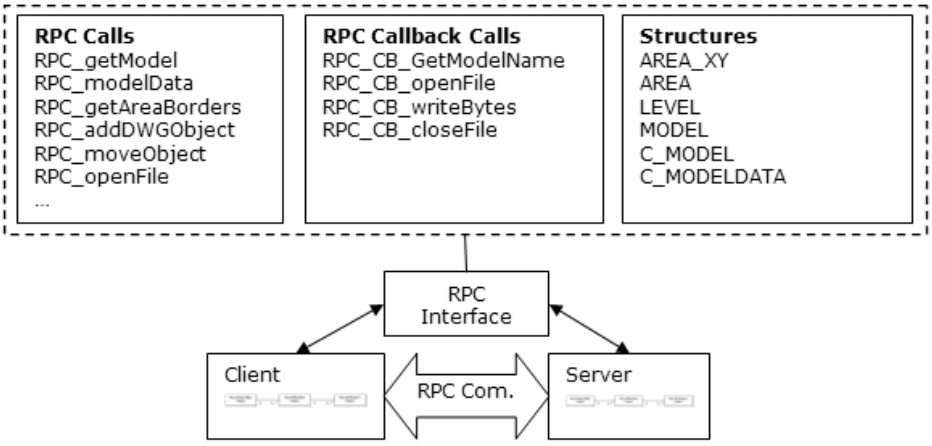

Figure 7.2. Overview of the IDL file.

Structure Name	Data types	Size
AREA_XY	200 double, 32 integer	1600 bytes
AREA	2 integer, 30 wchar_t	68 bytes
LEVEL	1 integer, 30 wchar_t	64 bytes
MODEL	70 wchar_t	140 bytes
C_MODEL	30 MODEL	4200 bytes
C_MODELDATA	1 MODEL, 100 LEVEL, 100 AREA, 3 integer	13352 bytes

Table 7.1. Structures.

RPC Calls	Data types	Size
RPC_getOpenModels	1 C_MODEL	13352 bytes
RPC_getModel	1024 wchar_t, 1 C_MODELDATA	15400 bytes
RPC_getActiveModel	1 C_MODELDATA	13352 bytes
RPC_modelData	1 integer, 1 C_MODELDATA	13356 bytes
RPC_getAreaBorders	2 integer, 1 AREA_XY	1608 bytes
RPC_getLevelDWG	3 integer	12 bytes
RPC_openFile	1 integer	4 bytes
RPC_writeBytes	2 integer, 32767 unsigned char	32775 bytes
RPC_closeFile	1 integer	4 bytes
RPC_addDWGObject	3 integer, 3 double	36 bytes
RPC_getObject	4 integer, 3 double	40 bytes
RPC_moveObject	2 integer, 3 double	32 bytes
RPC_addSharedParameter	2 integer, 2048 wchar_t	4104 bytes
RPC_setObjectParameter	2 integer, 2048 wchar_t	4104 bytes

Table 7.2. RPC Calls.

RPC Callback Calls	Data types	Size
RPC_CB_GetModelName	1024 wchar_t	2048 bytes
RPC_CB_openFile	1 integer	4 bytes
RPC_CB_writeBytes	2 integer, 32767 unsigned char	32775 bytes
RPC_CB_closeFile	1 integer	4 bytes

Table 7.3. RPC Callback Calls.

7.4 File Transfer

An implementation to send files between the client and server have been added. This is necessary when we like to send DWG drawing which is used to represent our objects and floor views. All binary transfer is made by function calls. The application protocol used starts by instantiating a buffer on the server side by calling a openFile function which will create a buffer specified. All data are transferred using function call with an ID and a 32KB argument. We are sending 32KB packages until the buffer have been filled. The buffer is written to disk when we close the transfer using a closeFile function call. Both server and client have an instance of these functions. When the server sends a file to the client it uses function calls. Figure 7.3 show the data transfer from client to server.

7.5 Class Mapping

Both client and server use a class hierarchy which is different in content but uses the same hierarchy. The content in the classes are represented by the model and objects in the Revit Architecture, which means that the coordinates are represented in the Revit Architecture coordinate system.

In Revit Architecture the user can have many models open at the same time. So to create an implementation that can handle many models and objects at the same time we can create multiple model classes but still use the same instance of

Figure 7.3. File Transfer from Client to Server.

the connection. The Revit Handler class which contains all code essential to the communication also contains one or more Model Classes. Then the Model Classes contains one or more Object Classes.

Each class has its unique id and other data that are essential to the synchronization and communication. When an object is sent from the client to the server, the server uses the unique id to know which class the object should be synchronized with.

7.6 Model Orientation

When objects are transferred we have to make a transformation between the coordinate systems. We have to use a scaling factor because Revit Architecture use feet and CET Designer use meter. The direction of x and y are the opposite and we also have to know how the objects are located to the origin in the other coordinate system.

As mentioned in chapter 5, a user who works in CET Designer would need a drawing where she sees all the borders from the rooms in Revit Architecture. A DWG drawing representing a floor plan is therefore transferred to CET Designer. The DWG drawing is then used as a reference between the coordinate systems. So when an object is transferred between the coordinate systems this DWG is used as a reference system. The client then transforms the coordinates to Revit Architecture coordinates before they are sent to the server.

7.7 Synchronization

All objects and model data that are stored in classes are in Revit Architecture coordinate system. This includes classes in both client and server. These classes are only updated when a synchronization is made or when a refresh is made. No data

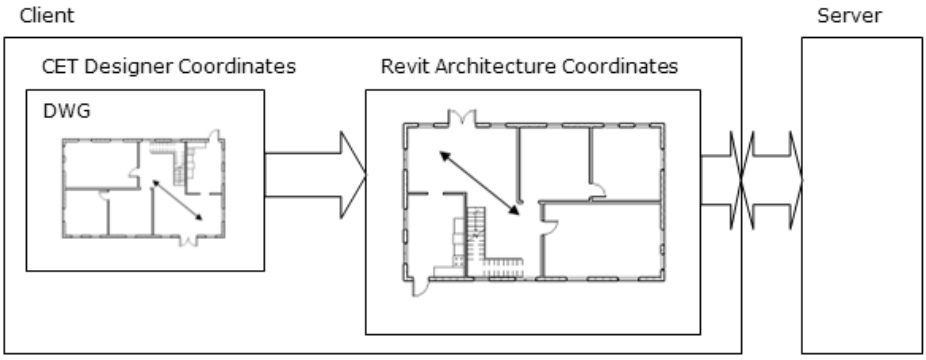

Figure 7.4. Overview of Model Transformation from Client to Server.

in the classes are referenced to the object information itself. When synchronization is made we have to extract the real information from both applications.

Unfortunately there is no way to add new objects in Revit Architecture that could be represented in CET Designer. So the synchronization is made so that each object that has been transported from CET Designer to Revit Architecture is compared with the current objects of CET Designer. The first version of the synchronization is very simple. The client asks if the object exists in Revit Architecture and then replaces it with the corresponding object in CET Designer. After the client has looped through all objects in CET Designer and the server has compared them with its corresponding objects the server knows which objects do not exist anymore, and easily deletes them from the model. This is a very simple synchronization which does much work on objects that has not changed. It is possible to create a more advanced synchronization with the existing implementation. This can be made by comparing more data, then we know if the objects have changed or not.

7.8 File Storage

The server has to store data about objects that have been transferred from the client. These data are stored to the disk when the server closes. Then we can continue the synchronization process at a later time. Data that are stored are object ids, DWG URLs, and parameters. C++ structures are used to store these data. The file contains a header structure which tells how much data the file contains and after the header there is one header for each data that is stored. Figure 7.5 shows how the file is structured.

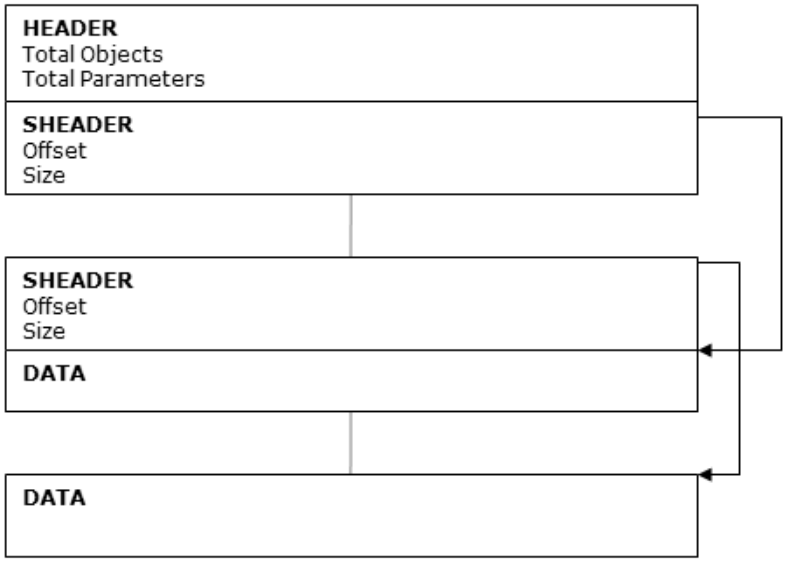

Figure 7.5. File Structure Overview.

Chapter 8

Measurements

In this chapter we will test the inter process connection, the time delays and if it is possible to work over long distances.

8.1 RPC Transfers

Here we test how long it takes to do certain communication actions in the program. The communication we use is Microsoft RPC with the IPC protocol "`ncacn np`" and the TCP/IP protocol "`ncacn ip tcp`". We will test if there is any difference if we use these protocols on a local computer and test how the response times are when doing TCP/IP over internet. The tests over internet were made between Stockholm and Linköping. The distance is approximately 200 kilometers.

All measures are based on the average of 100 sends in a row.

8.1.1 RPC Calls

We will measure how long time it takes for the client to make a RPC call to the server. The time is the time it takes from when the client has sent the RPC call till when the server has replied. See Table 8.1.

8.1.2 File Transfer - Client to Server

Here we test the file transfer between the client and server. Time is average of 10 sends. See Table 8.3.

8.1.3 File Transfer - Server to Client

Here we test the file transfer between the server and client. Time is average of 10 sends. See Table 8.4.

ncacn_np on local computer	Argument Size	Time
Time is the time for one call calculated from the average of 1000 RPC Calls. Argument size is the total size of all arguments.	0 Kbytes	0.024ms
	32 Kbytes	0.325ms
	64 Kbytes	0.509ms
	128 Kbytes	1.138ms
	256 Kbytes	2.177ms

ncacn_ip_tcp on local computer	Argument Size	Time
Time is the time for one call calculated from the average of 1000 RPC Calls. Argument size is the total size of all arguments.	0 Kbytes	0.052ms
	32 Kbytes	0.320ms
	64 Kbytes	0.491ms
	128 Kbytes	1.013ms
	256 Kbytes	1.960ms

ncacn_ip_tcp over internet	Argument Size	Time
Time is the time for one call calculated from the average of 100 RPC Calls. Argument size is the total size of all arguments.	0 Kbytes	12.02ms
	32 Kbytes	58.19ms
	64 Kbytes	360.6ms
	128Kbytes	731.8ms
	256Kbytes	1.460s

Table 8.1. RPC Calls.

ncacn_np on local computer	Argument Size	Time
Time is the time for one call calculated from the average of 1000 RPC Calls and 1000 Callbacks. Argument size is the total size of all arguments in one call or callback.	0 Kbytes	0.063ms
	32 Kbytes	0.545ms
	64 Kbytes	0.990ms
	128 Kbytes	2.100ms
	256 Kbytes	4.105ms

ncacn_ip_tcp on local computer	Argument Size	Time
Time is the time for one call calculated from the average of 1000 RPC Calls and 1000 Callbacks. Argument size is the total size of all arguments in one call or callback.	0 Kbytes	0.126ms
	32 Kbytes	0.638ms
	64 Kbytes	0.980ms
	128 Kbytes	2.018ms
	256 Kbytes	4.031ms

ncacn_ip_tcp over internet	Argument Size	Time
Time is the time for one call calculated from the average of 1000 RPC Calls and 1000 Callbacks. Argument size is the total size of all arguments in one call or callback.	0 Kbytes	24.58ms
	32 Kbytes	376.2ms
	64 Kbytes	721.2ms
	128 Kbytes	1.421s
	256 Kbytes	2.826s

Table 8.2. RPC Calls and Callbacks.

ncacn_np on local computer	File Size	Time
Sending File from Client to Server. File Size is the size of the buffer that is sent. Time is the time of one send calculated from the average of 10 sends in a row.	16Kb	2.652ms
	64Kb	4.395ms
	128Kb	7.735ms
	256Kb	15.90ms
	512 Kb	25.75ms
	1024Kb	53.16ms

ncacn_ip_tcp on local computer	File Size	Time
Sending File from Client to Server. File Size is the size of the buffer that is sent. Time is the time of one send calculated from the average of 10 sends in a row.	16Kb	3.486ms
	64Kb	5.311ms
	128Kb	8.241ms
	256Kb	16.11ms
	512 Kb	26.84ms
	1024Kb	52.85ms

ncacn_ip_tcp over internet, Linköping - Stockholm	File Size	Time
Sending File from Client to Server. File Size is the size of the buffer that is sent. Time is the time of one send calculated from the average of 5 sends in a row.	16Kb	76.02ms
	64Kb	177.4ms
	128Kb	279.5ms
	256Kb	487.4ms
	512 Kb	907.3ms
	1024Kb	1.734s

Table 8.3. File Transfer Client to Server.

ncacn_np on local computer	File Size	Time
Sending File from Server to Client. File Size is the size of the buffer that is sent. Time is the time of one send calculated from the average of 10 sends in a row.	16Kb	2.439ms
	64Kb	6.936ms
	128Kb	13.160ms
	256Kb	27.134ms
	512 Kb	51.391ms
	1024Kb	102.127ms

ncacn_ip_tcp on local computer	File Size	Time
Sending File from Server to Client. File Size is the size of the buffer that is sent. Time is the time of one send calculated from the average of 10 sends in a row.	16Kb	2.883ms
	64Kb	7.274ms
	128Kb	13.673ms
	256Kb	26.632ms
	512 Kb	52.843ms
	1024Kb	102.809ms

ncacn_ip_tcp over internet, Linköping - Stockholm	File Size	Time
Sending File from Server to Client. File Size is the size of the buffer that is sent. Time is the time of one send calculated from the average of 10 sends in a row.	16Kb	360.6ms
	64Kb	1.002s
	128Kb	1.652s
	256Kb	2.951s
	512 Kb	5.548s
	1024Kb	10.74s

Table 8.4. File Transfer Server to Client.

Chapter 9

Result Analysis

This chapter will discuss the result of the thesis.

9.1 Introduction

The main goal of the investigation was how a connection could be achieved between CET Designer and Revit Architecture. There were a few options how this could have been done. One of them was to use the worksharing feature that is included in Revit Architecture. This feature is used widely when users want to share their project with many other architects. Autodesk does not provide developers with the source code to handle their project files or other parts of their projects. If we could have used this way to communicate we could just let CET Designer connect to the file and we would not have any limitation for what we can do. That includes a live synchronization between the applications. The only way we could get information from the models was through the API that Autodesk let developers use to build extensions to Revit Architecture. This API has a few limitations to manipulate certain data like 3D representations.

9.2 Revit Architecture as Server

The choice to make Revit Architecture as server was not easy. From the beginning this was the first choice that came to the hand. But in the end it has showed to be a good choice. Revit API has a very well developed interface that lets us get, and manipulate data from an external application. The API is not guaranteed by Autodesk to work with multiple threads, the API can handle multiple threads for read but does not always work when manipulating data in the models. CET Designer does not let multiple threads work in their models. RPC is multithreaded, so by letting Revit Architecture act as server we can take benefit from it.

CET Designer will be the program that creates the objects that will be synchronized. In Revit Architecture the user can only do small changes that can show up as "drafts" for CET Designer to work on.

9.3 Multiple Threads

The RPC Server is creating a new thread for each function call that is made. Autodesk tells us that working with multiple threads can work in certain examples but is not guaranteed to work. This has forced us to store data in the memory and submit them with the main thread that is executed when the main thread is free. This could have been solved by letting the main thread listen to calls from another type of communication or let the function calls start events that will execute the main thread to do certain work. [19]

9.4 Object Presentation

The customer meetings were very informative. The customers' main goal with Revit Architecture is to have a way to represent their objects with much of the features that are available in CET Designer. Unfortunately we cannot create family objects with the current version of Revit API and therefore we cannot add object specific features such as stretch and rotate.

After some discussion with Autodesk there was an option where we could use family objects. It is possible to add DWG drawings in a family object and doing so we can add properties like rotation angle. Unfortunately we cannot manipulate the DWG drawing in the family by any other way than rotate. But it is possible to add properties that only families can have.

This is not exactly what the customers wanted but it is closer than just importing DWG objects directly into the model. This way is not used in the current version but it is certainly possible to add in a future version.

9.5 Limited RPC Calls

With the current implementation all function calls in the communication have arguments with fixed size. This could create problems with very big models that have many floors and areas. If we try to import data from a very big model, data will be lost.

A way to solve this would be to use callback functions to send these data to the client. The measurements in chapter 8 show that callback functions are fast in communication. Callback functions can replace the function return arguments without severe time loss.

9.6 Using serialized Classes

Another option to the file storage would be to use serialized classes. All objects are represented by classes and the parameters could also be stored in these. This would have been a much better way to store data when we do not need to change anything when the classes are changed.

9.7 Measurements

All measurements show that the time is linear in the size of the argument in the RPC Call.

RPC Calls are very fast when they are made on the local computer. Table 8.1, 8.2, 8.3, 8.3 show that there is no significant difference between the IPC and TCP/IP protocol when working on the local computer. So in a future version it could be a good choice to always use the TCP/IP protocol, both locally and over a network.

The function calls for the communication are very fast and are not the bottleneck when we like to transfer data between the applications. The time it takes to handle the arguments that are sent with the function call are about 4 times less than the time it takes to copy the argument to the buffer. Measurements from clean function calls show us that it would take about 11ms (9.1) to send a 1MB file from the client to server. The bottle neck in the function call is that the argument memory has to be copied between the memory spaces. The argument then has to be copied to the buffer where the file is stored. So the whole process to send a 1MB file could be optimized to take twice the time as the clean function calls would take. The current implementation takes about 50ms which probably could be optimized. The equation below show how long time it take to transfer 1MB of arguments with the "`ncacn ip tcp`" protocol. 0.35 is the time for a clean function call with 32KB argument. 32 are the total calls to send 1 MB. 0.05 is the time it takes to send an empty function call. The total of empty argument calls is 2.

$$(0.35 * 32) + (0.05 * 2) = 11.1ms \tag{9.1}$$

9.8 Future Improvements

Features that could be improved in current implementation are:

- The synchronization can be improved substantially.
- Objects can be represented by family DWG objects.
- RPC calls could be optimized

9.9 Summary

Here is a summary about what is possible and what projects could be done in the future.

9.9.1 Network Interaction between CET and Revit

We can exchange and manipulate data between the applications. The communication could be either on a local computer or over a network. The data could be stored locally or in a database.

9.9.2 CET Objects in Revit

It is possible to represent CET Designer objects as Nonparametric Family Types in Revit Architecture:

Nonparametric families do not provide changeable dimensions. They will only be one type and it is not possible to change the properties of it. (3.6.2)

Custom parameters can be added to the objects, such as cost, developer, and model id. These can be used with the schedule feature in Revit Architecture. Rendering material has to be added manually after the objects have been added. These could be added automatically if a Material Catalog is present in Revit Architecture.

Note: CET Designer objects in Revit Architecture will work like any imported family libraries beside the fact that it is only possible to duplicate, delete, move, and change parameters.

9.9.3 Revit Objects in CET

Any 2D views of floors and areas in Revit Architecture can be represented as DWG Drawings in CET Designer. Objects imported from CET Designer that have been duplicated in Revit Architecture can be represented as a copy of the CET Object in question. If Revit Architecture is supplied with a Family Catalog, the objects could be represented as drafts of the CET Object. The designer will then need to modify the CET Object to its true parameters in CET Designer.

9.9.4 Examples of Future Projects

Designer - Designer working with the interior solutions in CET Designer
Architect - Architect working in Revit Architecture

- **Create Revit Architecture Catalogs from CET Designer Objects** - A tool only present for the developers is used to create Nonparametric Family Objects in Revit Architecture. This Catalog can then be available for download on the manufacturer homepage.

- **Working with a Live Connection** - Designer and Architect start to share addresses and passwords to establish a live connection between their applications. After a live connection has been established the Designer requests all information about the building. The floors are represented in different alternative workspaces in CET Designer with DWG Drawings for each floor. Both Designer and Architect work with the model and when one of them likes to synchronize their model they can request to make a connection. When a connection is established for the second time the floor view is updated in CET Designer if the Architect has changed the building. Then the Designer can choose to synchronize her work in CET Designer to Revit Architecture. The Architect then gets the interior solutions as Nonparametric families in his model. The architect can then choose to move, duplicate or remove

objects of the interior designs and request for a third connection. The architect chooses to synchronize from Revit Architecture to CET Designer and the changes will be present in CET Designer.

- **Working Towards a Database** - The Architect starts by creating a new database for her project. The designer connects to the database and downloads all necessary data about the building. The floors are represented in different alternative workspaces in CET Designer with DWG Drawings. When the designer has an interior solution she uploads it to the database. The Architect can then choose, at any time, to download it to her model. The architect can choose to move, duplicate or remove objects of the interior designs and upload the changes to the database. Both the Designer and the Architect can upload and download data to the database at any time.

Bibliography

[1] Autodesk University (2005), *Basics of the Autodesk Revit Building API*, http://www.augi.com/education/auhandouts/2005/BD32-2.pdf (Accessed 2008-09-27).

[2] Autodesk (2008), *Building Information Modeling*, http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=9976276 (Accessed 2008-09-27).

[3] Configura (2008), *Configura CET Designer Help*, In-program help guide.

[4] Eddy Krygiel, Greg Demchak, Tatjana Dzambazova (2008), *Introducing Revit Architecture 2008 BIM for beginners*, WILEY PUBLISHING, INC.

[5] Eddy Krygiel, Greg Demchak, Tatjana Dzambazova (2008), *Mastering Revit Architecture 2009*, WILEY PUBLISHING, INC.

[6] Microsoft, *Microsoft Developer Network*, http://msdn.microsoft.com.

[7] Microsoft, *Compiling to MSIL*, http://msdn.microsoft.com/en-us/library/c5tkafs1(VS.71).aspx. (Accessed 2009-02-17).

[8] Microsoft, *Managed Extensions for C++ Programming*, http://msdn.microsoft.com/en-us/library/aa712574(VS.71).aspx. (Accessed 2009-02-17).

http://msdn.microsoft.com/en-us/library/aa712867.aspx

[9] Autodesk (2008), *Developer Center - Revit Architecture and Revit Structure*, http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=2484975 (Accessed 2008-09-27).

[10] Autodesk (2008), *Revit Architecture 2009 Families Guide*, http://www.autodesk.com.

[11] Autodesk (2008), *Revit Architecture 2009 User's guide*, http://www.autodesk.com.

[12] Autodesk (2008), *Revit Architecture Help*, In-program help guide.

[13] Autodesk (2008), *Revit 2009 SDK*, http://images.autodesk.com/adsk/files/revit2009sdk.zip (Accessed 2008-09-01).

[14] Autodesk (2008), *Revit 2008 API User Manual*, http://www.autodesk.com

[15] Runa Patel and Bo Davidson (2003), *Forskningsmetodikens Grunder*, Studentlitteratur AB

[16] Autodesk (2008), *Autodesk Homepage*, http://www.autodesk.com

[17] Autodesk (2008), *Autodesk ADN Support*, http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=723353

[18] Sune Rydqvist (2008), *CET*, The Strenght of PGC.

[19] Microsoft, *Multithreaded Apartments*, http://msdn.microsoft.com/en-us/library/ms693421(VS.85).aspx (Accessed 2008-11-01).